A FAST NON LOCAL MEANS ALGORITHM FOR IMAGE DENOISING

Anshuman Dwivedi

Guided by:

Dr Rathna G N

Table of Contents

The Academies' Summer Research Fellowship Programme 2019

A FAST NON LOCAL MEANS ALGORITHM FOR IMAGE DENOISING

Anshuman Dwivedi

Indian Institute Of Technology Patna

Guided by:

Dr Rathna G N

Principal Research Scientist,Department of Electrical Engineering

Indian Institute of Science Banglore

Abstract

Image Denoising is the method by which we preserve the details of an image and remove the noise as far as possible. It is used as algorithm for improving the quality of image as well as for many image processing applications like face recognition. **Non local means**(NLM) is an algorithm for image denoising which is conceptually simple but has high computational complexity. It replaces each pixel with weighted sum of all the pixels where weights are assigned on the basis of similarity between pixels. Due to computational comlexity there is a need to accelerate this algorithm. The search window can be restricted to reduce computational cost but as the algorithm is non local in nature we can not achieve optimum level of speed using this. **Fast Non Local Means** (FNLM) is an algortihm which reduces NLM complexity by using **summed area table** or **integral image**. Integral Image is an algorithm and data structure used for generating sum of values in any subset of a matrix quickly and efficiently. Using the integral image the summed value per pixel is computed in constant time. As the operations performed in this algorithm are independent of each other we can perform them in parallel using **FPGA**. FPGAs enable us to create architecture for parallel execution of the algorithm. The time complexity of NLM is dependent on number of pixels,patch radius and search radius where as for FNLM it only depends on number of pixels and search radius.

Keywords:*Image Denoising,Integral Image or Summed Area Table,FPGA and time complexity.*

Abbreviations

NLM	**Non Local Means**
FNLM	**Fast Non Local Means**
FPGA	**Field Programmable Gate Array**
SNLM	**Separable Non Local Means**
SFNLM	**Super Fast Non Local Means**
SDSS	**Sum of Difference Squared Signal**
SDSI	**Sum of Difference Squared Image**

1 INTRODUCTION

1.1 Background

Image noise is a variation of random nature in the brightness or color information of image. **Image Denoising** is the process of removal of noise while preserving the details of image. Consider a noisy pixel np=p+n where p is true pixel value and n is noise added to that.Averaging pixels in neighborhood results in removal of noise as it's mean is zero.So idea is simple,we need to have set of similar pixels to average out noise.See an example below:

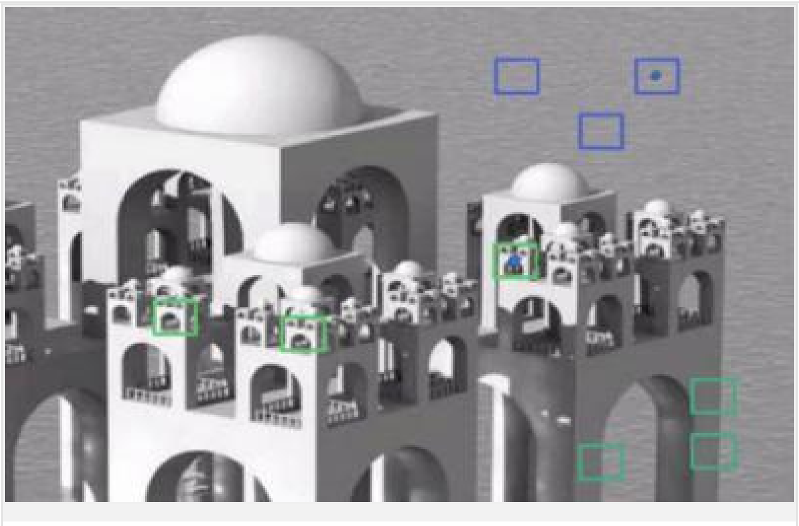


Fig I explaining_nlm

The patches marked with same color look similar.The idea is to take a patch centered around a pixel and search for patches similar to it and assign weight accorrding to similarity and take weighted average of them and assign it to the pixel.

Due to non local nature algorithm given above is known as **Non Local Means**(NLM). In this work we implement given algorithm.As it involves searching the entire image for similar pixels thus giving rise to high computational cost. We try to reduce the computational cost by searching a large neighborhood instead of entire image and thus making it semi-local in nature.Further reduction in time complexity can be brought by using algorithms such as **Fast Non Local Means** (FNLM) and **Separable Non Local Means** (SNLM) which are implemented in this work.

1.2 Problem Statement

To reduce the computational cost of NLM algorithm which is used for image denoising.

1.3 Objective

Implementation of various algorithms to minimize the computational cost of Non local means.

1.4 Scope

FPGA implementation of these algorithms can reduce the computational cost further as operations used in given algorithms are independent in nature.

2 LITERATURE REVIEW

1]Non Local Means algorithm

In 2005,**A Buades**,**B Coll and JM Morell** proposed a non local means algorithm for image denoising.It is based on averaging pixels based on similarity between pixels. The mathematical formulation is as follows :

Given a discrete noisy image, denoised pixel value is measured by taking weighted average of all the pixels in the image, where weights are assigned on the basis of similarity between pixels.

$$dp(i) = \sum_{j\epsilon I} w(i,j)p(j) \quad (1)$$

$$w(i,j) : weight\ assigned\ to\ pixels,$$

$$p(j) : pixel\ value$$

$$dp(i) : denoised\ pixel\ value$$

Definition of weight: Weight is defined on the basis of similarity between two sqaure neighborhood centered at i and j respectively.

$$d(i,j) = ||p(N(i)) - p(N(j))||_{2}^{2}{}_{,\sigma} \quad (2)$$

$$N(k) : neighborhood\ centered\ around\ k$$

$$d(i,j) : euclidean\ distance\ between\ neighborhood\ of\ pixels$$

$$w(i,j) = \frac{e^{-\frac{D}{h^2}}}{TW(i)} \quad (3)$$

$$TW(i) = totalweight$$

$$h : filter\ parameter$$

Constraint on weight:

$$0 \leq w(i,j) \leq 1$$

$$\sum_j w(i,j) = 1$$

2] Fast Non local Means algorithm

Darbon,Jrme,et al. proprosed a method of efficiently computing the weigths by using a famous algorithm and data structure known as **summed area table or integral image**. The proposed mathematical formulation for one dimensional signal is as follows: Let us consider a signal S containing N values such that Ω ϵ [0,N-1].As NLM requires summation of squared difference between neighborhood pixels (eqn 2) thus we calculate the integral image of difference squared image.

$$SDSS_{dx}(p) = \sum_{k=0}^{p}(S(k) - S(k+dx))^2 \quad p \in \Omega \ (4)$$

Here p represents the neighborhood defined in eqn(2).Translating eqn(1) for one dimensional signal we get

$$d(i,j) = \sum_{t\in\Delta}(S(i+t) - S(j+t))^2 \quad \Delta \in [-p,p] \ (5)$$

using eqn(4) we can transform eqn(5) as

$$d(i,j) = SDSS_{dx}(i+p) - SDSS_{dx}(i-p) \ (6)$$

$$dx = j - i$$

Thus,if SDSS is known than we can calculate weight in constant time.Translation of above one dimensional algorithm for image is done in this work.

3]Separable Non Local Means algorithm or Super Fast Non Local Means algorithm

Sanjay Ghosh *and* *Kunal N. Chaudhury* proposed this algorithm which uses separable kernel.Since,kernel in NLM is not separablei.e results are different depending on whether we process row or column.They propose to take the optimum sum of both the results(processing row first and column second and vice-versa). The mathematical formulation of above algorithm is as follows:

Let us consider one dimensional signal S = {S(1),S(2)...,S(n)}.Since, for nlm we calculate d(i,j) which is given by eqn(5).Expanding eqn(5) we get

$$d(i,j) = \sum_{t=-p}^{t=p} S^2(i+t) + \sum_{t=-p}^{t=p} S^2(j+t) - 2\sum_{t=-p}^{t=p} S(i+t)S(j+t)$$

$$Let\ F(x,y) = S(x)S(y)$$

$$d(i,j) = \sum_{t=-p}^{t=p} F(i+t,i+t) + \sum_{t=-p}^{t=p} F(j+t,j+t) - 2\sum_{t=-p}^{t=p} F(i+t,j+t)$$

$$Let\ SF(i,j) = \sum_{t=-p}^{t=p} F(i+t,j+t)$$

$$d(i,j) = SF(i,i) + SF(j,j) - 2SF(i,j) \quad (7)$$

As it can be seen from eqn(5) we can calculate d(i,j) in constant time if we know SF. Proposed time complexities are tabulated below:

Table 1 **Complexity of above algorithms . delta_window and delta denote the radius of search window and the radius of patch, respectively, and N × N is the image size**

NLM	$O(N^2 delta\_window^2 delta^2)$
FNLM	$O(N^2 delta\_window^2)$
SNLM/SFNLM	$O(N^2 delta\_window)$

3 METHODOLOGY

Table 2 **Terms used in implementation of above algorithms**

windowSize/search_window	size of search window
patchSize/patch_window	size of square neighborhood
delta_window/search_radius	radius of search window
delta/patch_radius	radius of sqaure neighborhood
h	Filter Parameter
NI	Noisy Image
PI	Padded Image
DI	Denoised Image
I	Integral Image
TW	Total Weight
AW	Average Weight
σ	Noise variance

1 IMPLEMENTATION OF NLM:

1.1 Padding and it's use :

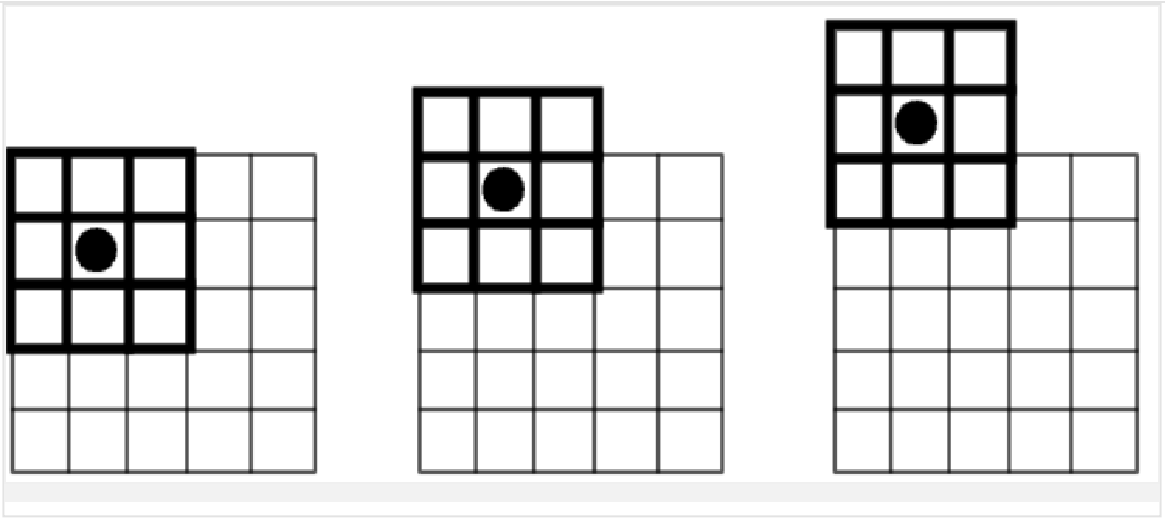


Fig II figure illustrating need of padding to meet boundary conditions

In this algorithm we have used **reflective padding** as it shows better results for boundary pixels as compared to other methods.The size of pad is equal to size of delta.

1.2 Psuedocode :

```
INPUT : NI,delta,delta_window
OUTPUT : DI
(original_row,original_col) <-- image size
PI <-- NI padded with delta

for row = delta to delta + original_row do
    for col = delta to delta + original_col do
        TW<--0
        AW<--0
        LPR<--max(-delta_window,delta-row)
        RPR<--min(delta_window,original_row+delta-row)
        LPC<--max(-delta_window,delta-col)
        RPC<--min(delta_window,original_col+delta-col)
        for i = LPR to RPR do
            for j = LPC to RPC do
                dist<--compute distance between two patches according to eqn(2)
                w(i,j)<--assign weight according to eqn(3)
                TW=TW+w(i,j)
                AW=AW+w(i,j)*I[row+i][col+i]
            end j
        end i
```

```
 DI[row][col] <--AW/TW
 end col
end row
```

Code 1 PSEUDOCODE OF NLM

2] IMPLEMENTATION OF FNLM

1.1] Padding In this algorithm we have used **reflective padding** as it shows better results for boundary pixels as compared to other methods.The size of pad is equal to size of delta+delta_window+1.

1.2]Psuedocode This implementation contains three modules: **a)INTEGRAL IMAGE CALCULATION** For an image NI,the value at any pixel in integral image I is given by

$$I(x,y) = \sum_{x' \le x, y' \le y} NI(x',y') \quad (8)$$

Integral image can be computed efficiently in a single pass over the image using equation given below:

$$I(x,y) = NI(x,y) + I(x-1,y) + I(x,y-1) - I(x-1,y-1) \quad (9)$$

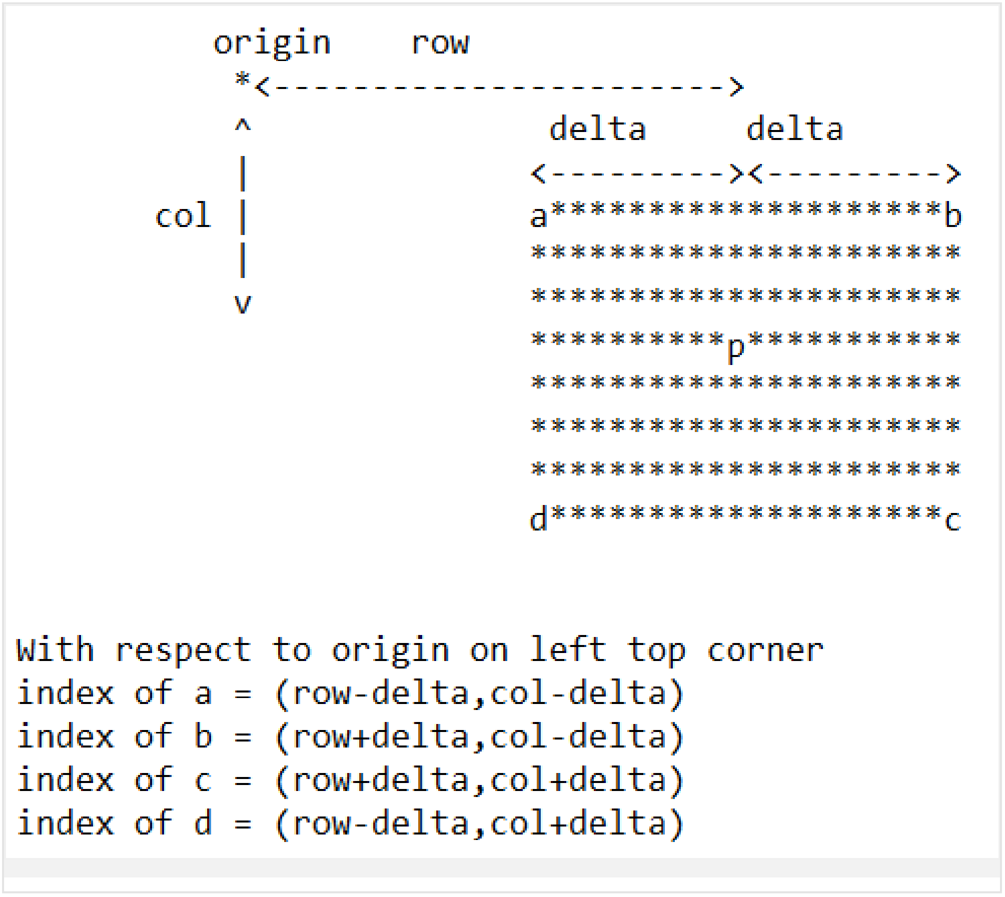
According to algorithm we need to calculate SDSI corresponding to SDSS(eqn 4) which can be computed efficiently in a single pass over the image using equation given below :

$$SDSI(x,y) = distance + SDSI(x-1,y) + SDSI(x,y-1) - SDSI(x-1,y-1) \quad (10)$$

```
INPUT : PI,t_row(translated row),t_col(translated col),σ
OUTPUT : SDSI between PI and same PI shifted by t_row and t_col
(n_row,ncol) <-- PI size
for row = max(1,-t_row) to min(n_row,n_row-t_row) do
 for col = max(1,-t_col) to min(n_col,n_col-t_col) do
 difference <-- PI(row,col) - PI(row+t_row,col+t_col)
 distance <-- difference*difference - σ*σ
 SDSI(row,col) <-- distance + SDSI(row-1,col) +SDSI(row,col-1) +
8. SDSI(row-1,col-1) : recursive step (eqn 10)
```

Code 2 CALCULATING SDSI

b)INTEGRAL IMAGE TO d(i,j)(see eqn 5) CONVERSION



```
           origin    row
            *<------------------------>
            ^              delta     delta
            |             <--------->><--------->
        col |             a********************b
            |             **********************
            v             **********************
                          **********p***********
                          **********************
                          **********************
                          **********************
                          d********************c


With respect to origin on left top corner
index of a = (row-delta,col-delta)
index of b = (row+delta,col-delta)
index of c = (row+delta,col+delta)
index of d = (row-delta,col+delta)
```

Fig III explaining conversion

```
1. INPUT : SDSI,h
2. OUTPUT : d(i,j)
3. d(i,j) <-- SDSI(a) + SDSI(c) - SDSI(b) -SDSI(d)
```

```
d(i,j) <-- max(d(i,j),0)/h*h
```

Code 3 CALCULATING d(i,j)

c)DENOISING STEP

```
//merging module 1 and module 2
INPUT : NI,delta,delta_window,h,σ
OUTPUT : DI
(row,col) <-- PI size
PI <-- NI padded with delta +delta_window + 1
WM(Weight matrix) <-- zero matrix with same size as PI
RM(Result matrix) <-- zero matrix with same size as PI
OI(Output image) <-- zero matrix with same size as NI
for t_row = -delta_window to delta_window do
 for t_col = -delta_window to delta_window do
 calculate integral image using module 1 do
 LPR <-- max(delta,delta-t_row)
 RPR <-- min(row-delta,row-delta-t_row)
 LPC <-- max(delta,delta-t_col)
 RPC <-- min(col-delta,col-delta-t_col)
 for i = LPR to RPR do
 for j = LPC to RPC do
 calculate d(i,j) using module 2
 //denoising step
 w(i,j) <-- calculate using eqn 2
 WM(i,j) <-- WM(i,j) + w(i,j)
 WM(i+t_row,j+t_col) <-- WM(i+t_row,j+t_col)
 RM(i,j) <-- RM(i,j)+w(i,j)*PI(i+t_row,j+t_col)
                     RM(i+t_row,j+t_col) <-- RM(i+t_row,j+t_col)+w(i,j)*PI(i,j)
 end j
 end i
 end t_col
end t_row
// normalizing step
for r = delta to row - delta do
       for c = delta to col - delta do
                    OI(r-delta,c-delta) <-- RM(r,c)/WM(r,c)
 end c
```

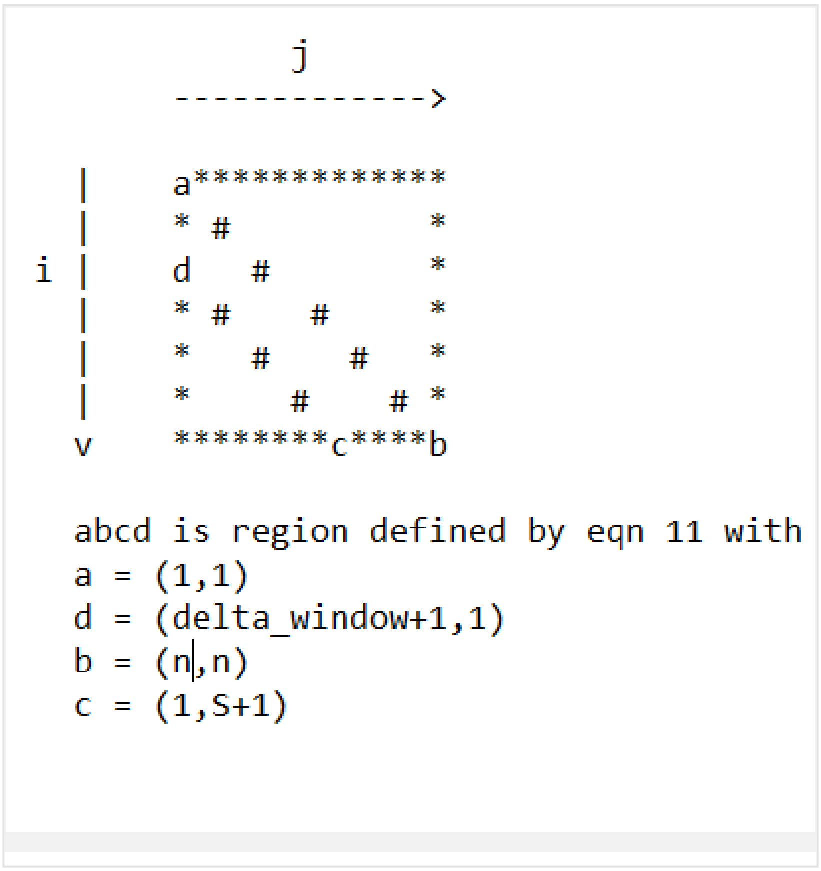
```
    34. end r
```

Code 4 DENOSING STEP

3]IMPLEMENTATION OF SNLM/SFNLM(for 1 dimensional signal)

As it can be seen from eqn 7 that it is symmetrical thus making the values of (i,j) to be iterated as:

$$(i, j) : 1 \leq i, j \leq n, i \geq j, |i - j| \leq delta\_window \quad (11)$$



```
                j
         ------------->

    |    a**************
    |    * #           *
  i |    d   #         *
    |    * #    #      *
    |    *   #    #    *
    |    *     #    #  *
    v    ********c****b

    abcd is region defined by eqn 11 with
    a = (1,1)
    d = (delta_window+1,1)
    b = (n,n)
    c = (1,S+1)
```

Fig IV region defined by eqn 11

Let us define u as another variable defined as 1 when diagonal is dc and as delta_window+1 when diagonal is ab and v as another variable which indexes elements on diagonals.

$$1 \leq u \leq delta\_window \; and \; 1 \leq v \leq n - delta\_window - 1 + u$$

Using transformation, we get

$$j = v \; and \; i = delta\_window + 1 - u + v$$

Now we fill the values in the region in array(F) according to our transformation with delta zeros between each diagonal.We fill zeros to remove chances of interference between diagonal values during filtering operation.

Thus index in array corresponding to value at (i,j) is given by

$$index_{ij} = (u - 1)\,delta + \sum_{z=1}^{u-1} (n - delta\_window - 1 + z) + v$$

using index total length of array comes to be

$$total\ length = (delta\_window + 1)\left(n - \frac{delta\_window}{2}\right) + (delta)(delta\_window)$$

using eqn 7 and array SF we get

$$d(i,j) = SF(i,i) + SF(j,j) - 2 * SF(i,j) \; if \; i \geq j \quad (12)$$

$$d(i,j) = SF(i,i) + SF(j,j) - 2S(j,i) \quad if \; j > i \quad (13)$$

```
1. INPUT : signal S :{S(1),S(2).....S(n)}, kernel B(BOX kernel) , delta ,delta_window
2. OUTPUT : array F
3.
4. F <-- array of size total length
5.
6. for u = 1 to delta_window +1 do
7.  for v=1 to n - delta_window-1+u do
8.  F <-- push back S(i)S(j) in F
```

```
     for i =1 to delta do
     F <-- push back 0 in F
     end i
     end v
    end u
    SF <-- F*B //this is the convolution step

    for i = 1 to n do
     TW <-- 0
     AW <-- 0
     for j = i-delta_window to i+delta_window do
     calculate d(i,j) using eqn 12 and eqn 13
     calculate w(i,j)
     AW <-- AW + S(j)*w(i,j)
     TW <-- TW + w(i,j)
     end j
    S(i) <-- AW/TW
    end i
```

Code 5 SNLM/SFNLM

4 RESULTS AND DISCUSSION

NOTE : ALL THE RESULTS ARE OBTAINED BY USING STANDARD IMAGES AND ADDING GAUSSIAN NOISE TO IT.

1]NLM

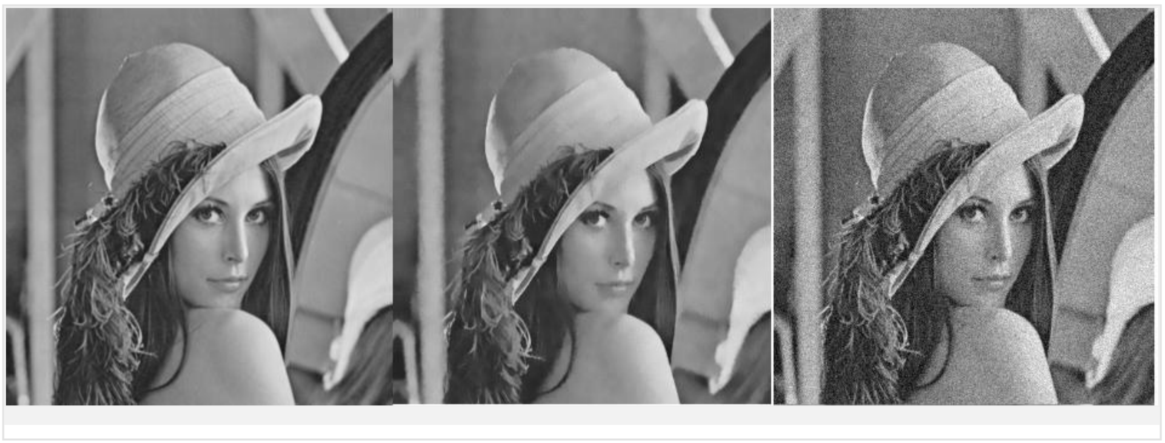


Fig V left - > original image, middle - > denoised image and right -> noisy image

Fig V shows the result for implementation of NLM with noise variance as 20 and filter parameter as 25 on lena image.The code was written in python where it took **1156.8** seconds to implement for **262,144**(512 x 512) pixel values with windowSize as **13** and patchSize as **5**.

The PSNR(Peak Signal To Noise Ratio) obtained was **29.26052**.

2]FNLM

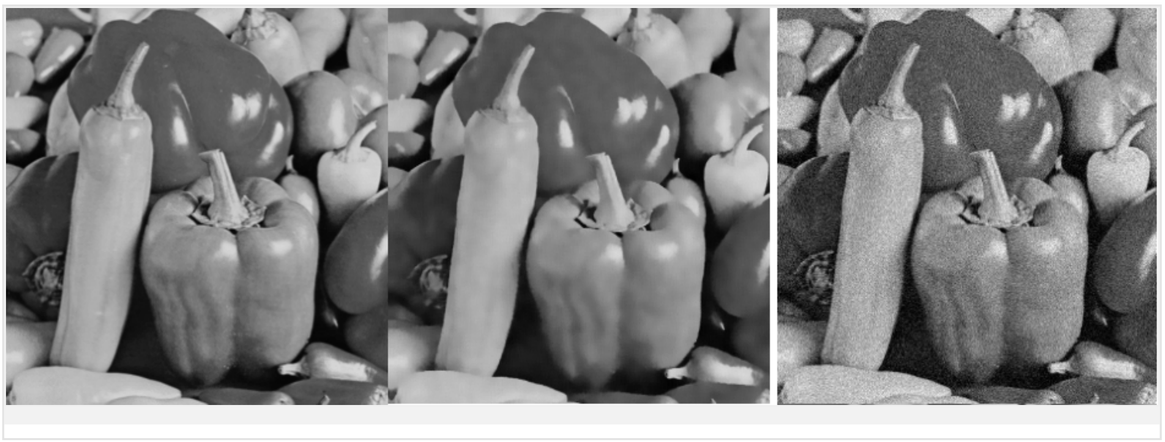


Fig VI left - > original image, middle - > denoised image and right -> noisy image

Fig VI shows the result for implementation of NLM with noise variance as 20 and filter parameter as 25 on pepper image.The code was written in python where it took **301.7** seconds to implement for **262,144**(512 x 512) pixel values with windowSize as **13** and patchSize as **5**.

The PSNR(Peak Signal To Noise Ratio) obtained was **28.83032**.

3] SNLM/SFNLM

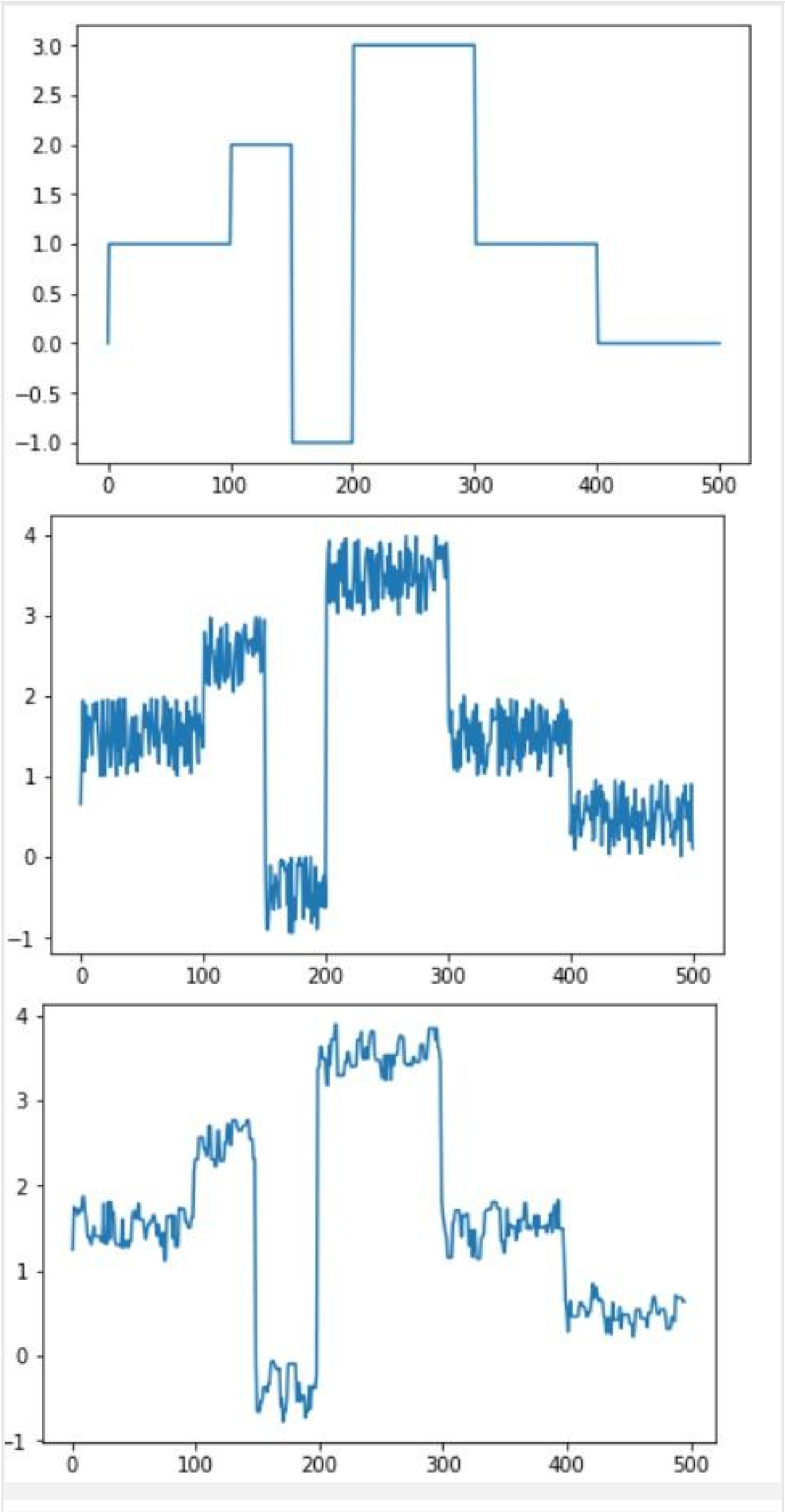


Fig VII Result are shown for implementation of SNLM/SFNLM on 1-d signal with random noise added to it

Fig VII shows the result for implementation of SNLM/SFNLM on 1-d signal with random noise and filter parameter as **10** and n as 500.

Dependence of noise variance and filter parameter (for FNLM) :

Table 3 PSNR vs h for noise variance as 20

standard images	h=15	h=16	h=17	h=18	
lena	31.748771410401318	32.310731993520974	31.74297376695972	31.295636782330533	3
pepper	31.175994659676533	31.397048462476018	31.035422005469393	30.744226273187493	30.
home	32.419062755528245	33.86901733767483	33.59910601066526	33.36097628522325	33.
cameraman	31.310043617157625	31.568530138197435	31.42070096123178	31.302321311177618	31

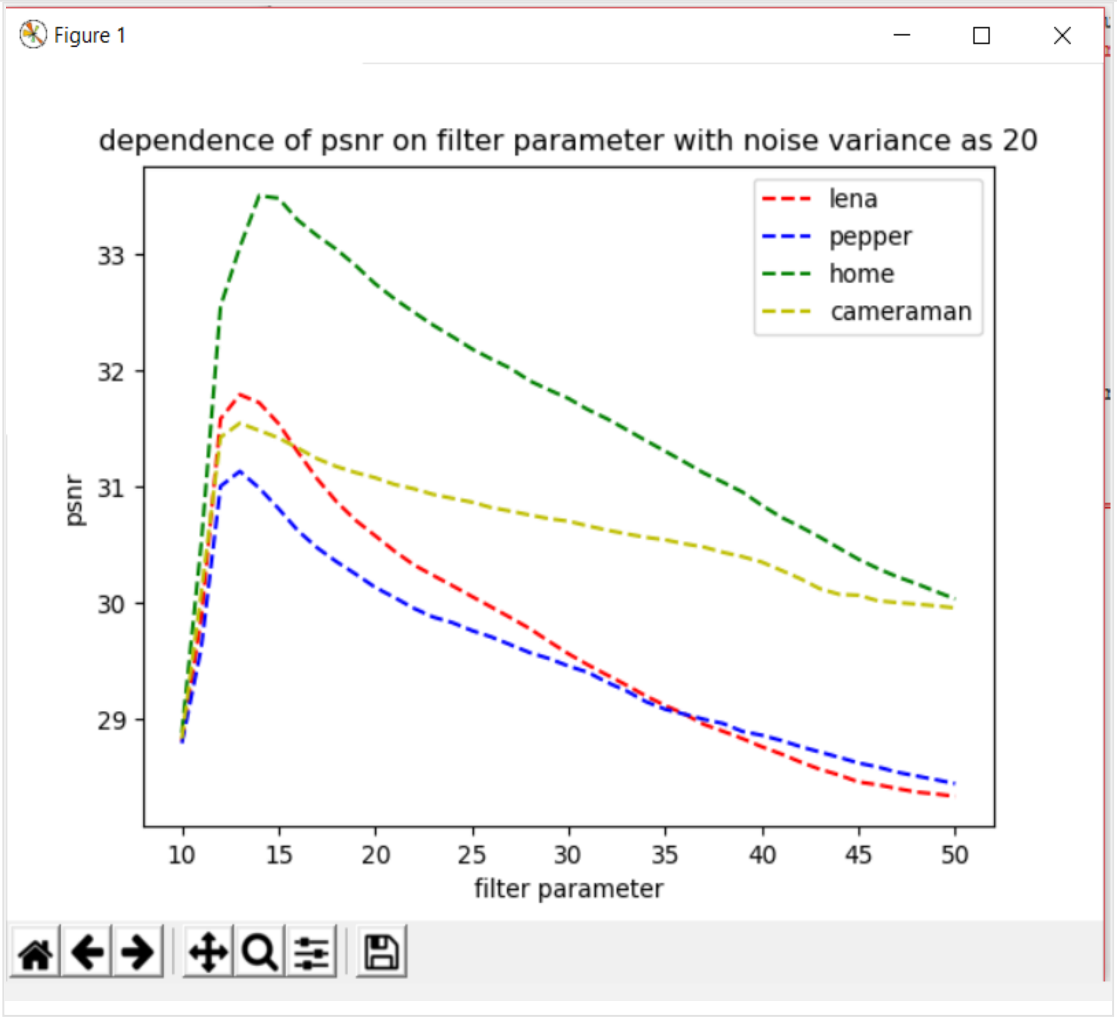


Fig VIII dependence of psnr on filter parameter with noise variance as 20.Plot is generated using matplotlib module of python

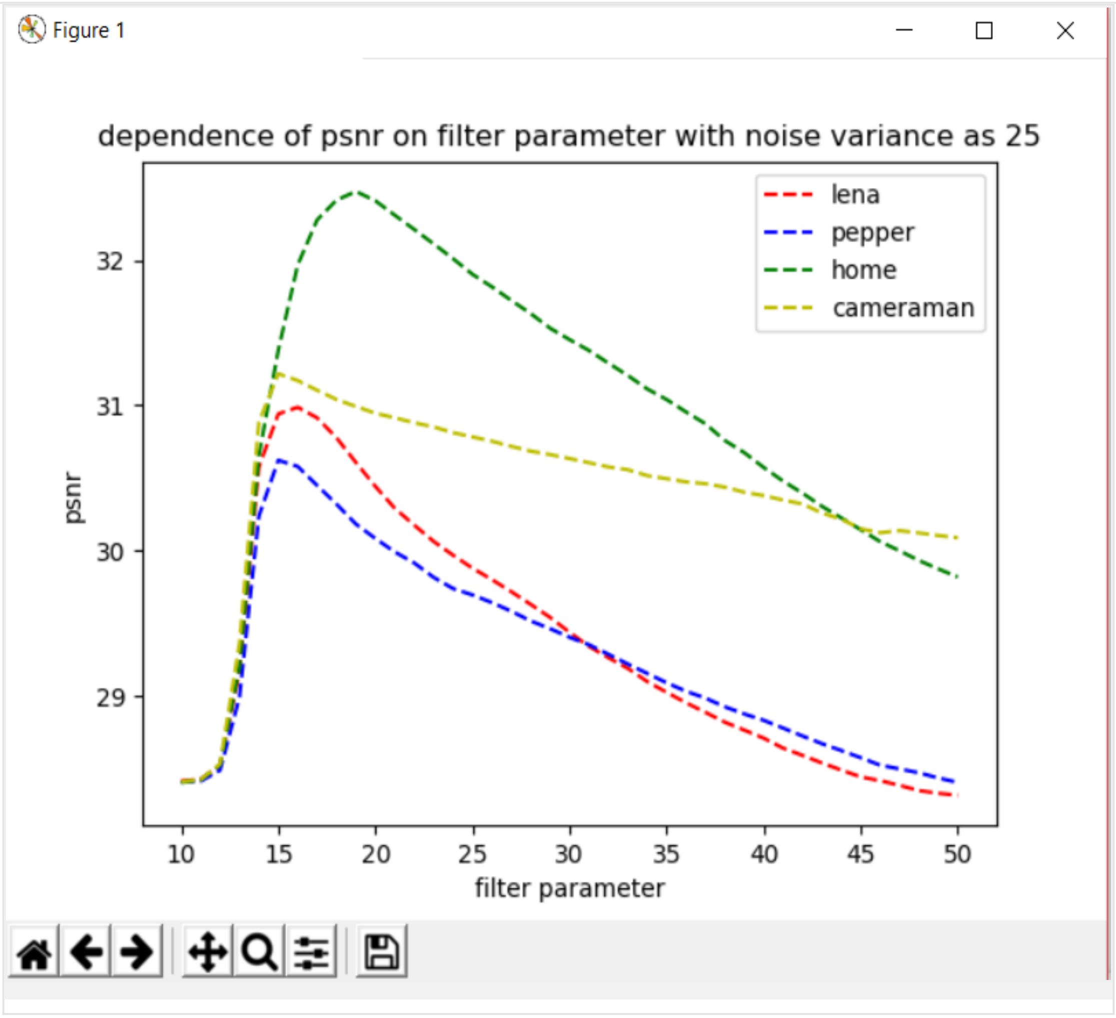


Fig IX dependence of psnr on filter parameter with noise variance as 25.Plot is generated using matplotlib module of python

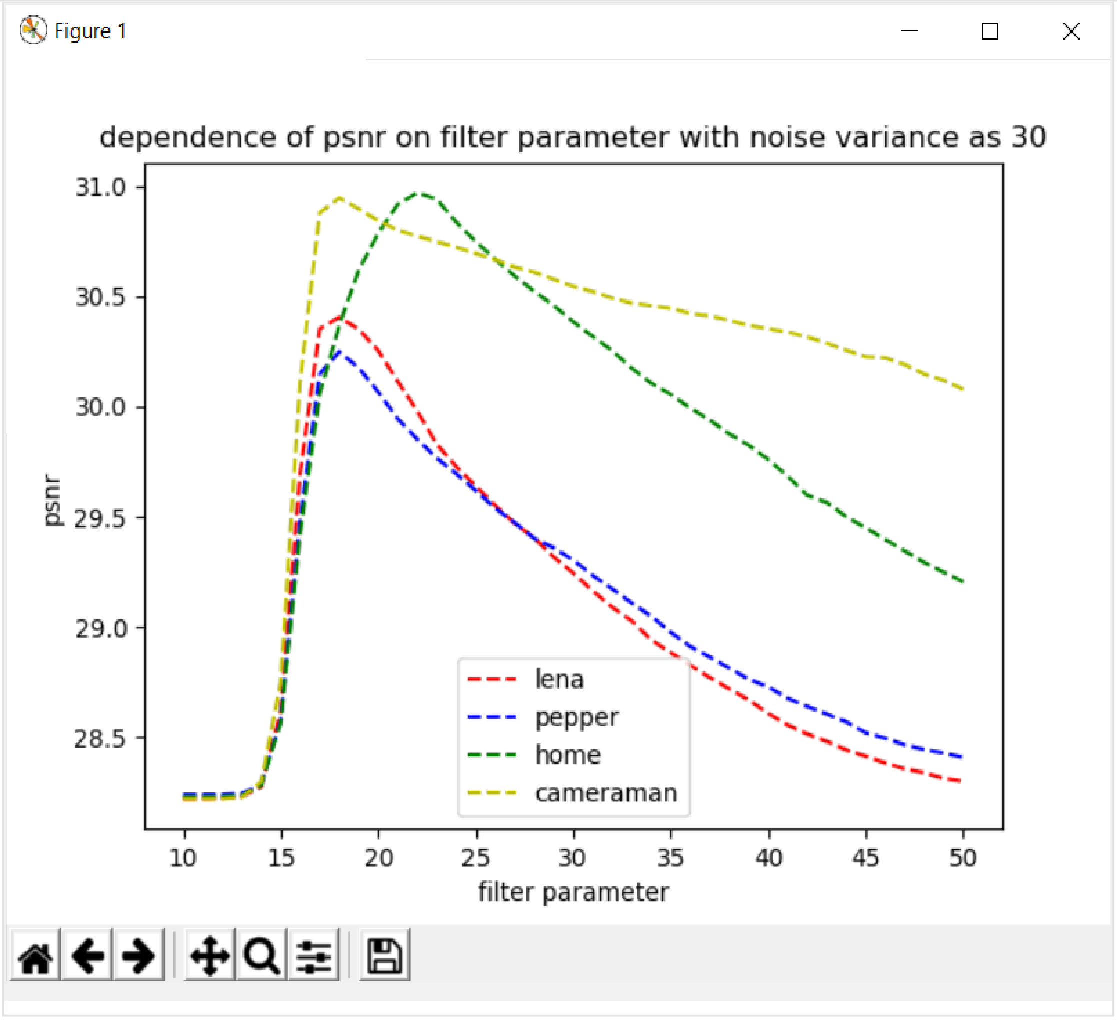


Fig X dependence of psnr on filter parameter with noise variance as 30.Plot is generated using matplotlib module of python

As it can be seen from above plots that the relation between filter parameter and psnr is approximetly bell shaped with peak obtained as follows:

Table 4 **Analysis of relation between SIGMA and h**

SIGMA(*noise variance*)	**h**(*filter parameter*)
20	$14 \leq h \leq 17$
25	$16 \leq h \leq 18$
30	$18 \leq h \leq 21$

As it can be seen with increasing noise variance,filter parameter for which PSNR is maximum is shifted towards right.The relation is almost linear between filter parameter and noise.The result were bit shifted more for home.

5 CONCLUSION

This work included implementation of an algorithm for image denoising named as Non Local Means.We reduced the computational cost of it by implementing algorithms such as FNLM and SFNLM/SNLM.The work was implemented in python using libraries numpy and opencv.We found relation between filter parameter and noise variance which was almost linear.

As operations involved in above algorithms are independent in nature,FPGA implementation is possible.

ACKNOWLEDGEMENTS

I am grateful to Academy's Summer Research Fellowship programme for giving me a chance to work at IISC.I would also like to thank my guide,Dr GN Rathna for her valuable advice and support throughout this project.I would also like to thank Mr Naveen Kumar for helping in the minute details of the project.I would also like to thank my family and friends for continuous support.

REFERENCES

1. J. Darbon et al., “Fast nonlocal filtering applied to electron cryomicroscopy,” in Proc. IEEE Int. Symp. on Biomedical Imaging, pp. 1331–1334 (2008).
2. Antoni Buades, Bartomeu Coll, Jean-Michel Morel, 2011, Non-Local Means Denoising, Image

Processing On Line, vol. 1
3. Jerome Darbon, Alexandre Cunha, Tony F. Chan, Stanley Osher, Grant J. Jensen, 2008, Fast nonlocal filtering applied to electron cryomicroscopy, 2008 5th IEEE International Symposium on Biomedical Imaging: From Nano to Macro
4. Sanjay Ghosh, Kunal N. Chaudhury, 2016, Fast separable nonlocal means, Journal of Electronic Imaging, vol. 25, no. 2, pp. 023026
5. Franklin C. Crow, 1984, Summed-area tables for texture mapping, Proceedings of the 11th annual conference on Computer graphics and interactive techniques - SIGGRAPH '84

Source

1. Fig I: https://www.coursera.org/course/images
2. Fig II: https://www.divilabs.com/2014/12/padding-borders-of-image-with-zeros-or.html